Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Dr. Hanjo Täubig
Tobias Lieber

Sommersemester 2011
Übungsblatt 4
1. Juni 2011

Grundlagen: Algorithmen und Datenstrukturen

Abgabetermin: In der jeweiligen Tutorübung

Hausaufgabe 1

Implementieren Sie die Methoden `insert`, `remove` und `find` für Hashing mit Linear Probing. Verwenden Sie die Hashfunktion

$$h(x) = ax \bmod m.$$

Stellen Sie sicher, dass nach dem Löschen eines Elements die Invariante gilt, dass für jedes Element e in der Hashtabelle mit idealer Position $i = h(\texttt{key}(e))$ und aktueller Position j gilt: $T[i], T[i+1], ..., T[j]$ sind besetzt.
Verwenden sie für Ihre Implementierung die auf der Übungswebseite bereitgestellten Klassen und verändern Sie für Ihre Implementierung *ausschließlich* die Klasse `LinHash`.
Achten Sie bei der Abgabe Ihrer Aufgabe darauf, dass Ihre Klasse `LinHash` heißt und auf den Rechnern der Rayhalle (`rayhalle.informatik.tu-muenchen.de`) mit der bereitgestellten Datei `main_h` kompiliert werden kann. Anderenfalls kann eine Korrektur nicht garantiert werden. Achten Sie darauf, dass Ihr Quelltext ausreichend kommentiert ist.
Schicken Sie die Lösung per Email mit dem Betreff `[GAD] Gruppe <Gruppennummer>` an ihren Tutor.

Hausaufgabe 2

Implementieren Sie die Methoden `insert`, `remove` und `find` für Cuckoo-Hashing.
Als Hashfunktionen werden Funktionen vom Typ

$$\left(\left(\sum_{j=0}^{k-1} a_j x^j\right) \bmod p\right) \bmod n$$

mit einem Vektor $a = (a_0, ..., a_{k-1})$ und ganzen Zahlen k, p und n verwendet, die bei der Initialisierung übergeben werden. Beachten Sie, dass x hier nur eine ganze Zahl und *kein* Vektor ist. Die Zahl n gibt hier die Größe der Hashtabelle an. Außerdem soll bei der Initialisierung der Hashtabelle ein Wert `max` übergeben werden, der einen Höchstwert für die Anzahl der Verschiebungen angibt (in der Vorlesung sind dies $2 \log n$). Wenn dieser Wert der Verschiebungen erreicht wird, so dürfen Sie das Programm sofort beenden.
Verwenden Sie für Ihre Implementierung die auf der Übungswebseite bereitgestellten Klassen und verändern Sie für Ihre Implementierung *ausschließlich* die Klasse `DynHash`.
Achten Sie bei der Abgabe Ihrer Aufgabe darauf, dass Ihre Klasse `DynHash` heißt und auf den Rechnern der Rayhalle (`rayhalle.informatik.tu-muenchen.de`) mit der bereitgestellten Datei `main_d` kompiliert werden kann. Anderenfalls kann eine Korrektur nicht garantiert werden. Achten Sie darauf, dass Ihr Quelltext ausreichend kommentiert ist.

Schicken Sie die Lösung per Email mit dem Betreff `[GAD] Gruppe <Gruppennummer>` an Ihren Tutor.

Aufgabe 1

Konstruieren Sie eine statische, perfekte Hashtabelle für die Elemente:

$$\begin{array}{cccccc} (16,10,11) & (8,2,15) & (7,12,8) & (1,10,3) & (13,11,14) & (6,11,14) \\ (7,3,16) & (2,2,8) & (10,5,15) & (7,3,14) & (2,10,1) & (14,11,6) \end{array}$$

Jedes Element x besteht aus den Stellen (x_0, x_1, x_2). Verwenden Sie jeweils passend eine der Hashfunktionen:

$$\begin{array}{c} (\sum_{i=0}^{2} 2^i x_i) \bmod 17 \\ (\sum_{i=0}^{2} a_i x_i) \bmod 7 \text{ mit } \mathbf{a} = (0,0,1) \text{ oder } \mathbf{a} = (6,6,2) \\ (\sum_{i=0}^{2} a_i x_i) \bmod 3 \text{ mit } \mathbf{a} = (1,0,0) \text{ oder } \mathbf{a} = (0,2,2). \end{array}$$

Aufgabe 2

Wir betrachten ein Negativbeispiel für eine Hashfunktion, die auf einem String s der Länge n aus Charactern $(s_1, \ldots, s_n)$ arbeitet:

$$h(s) := \sum_{i=1}^{n} \text{Anzahl der Einsen in der Binärdarstellung von } s_i.$$

Nehmen Sie an, dass jedes der 256 Zeichen in dem Text gleichwahrscheinlich vorkommt. Begründen Sie, warum Sie die Hashfunktion nicht für geeignet halten.

Aufgabe 3

Ein Hotelmanager hat n Buchungen für die nächste Saison. Sein Hotel hat k identische Räume. Die Buchungen enthalten ein Ankunfts- und ein Abreisedatum. Er will herausfinden, ob er zu allen Zeiten genügend Räume für die Buchungen zur Verfügung hat. Entwickeln Sie einen Algorithmus, der dieses Problem in Zeit $\mathcal{O}(n + \text{sort}(n))$ löst.